Wireless Mesh Interconnected Network

Eddie Penta
Department of Computer Science and Networking
Wentworth Institute of Technology
Boston, MA 02115, USA
pentae@wit.edu

***Abstract*— There are many existing consumer mesh network solutions that try to aim to unify an already existing LAN. However, there's no existing mesh network solution that aims to easily create a wireless mesh interconnected network (internet). In this paper, we present a concept and protocol that allows multiple LAN networks to form a VPN using a wireless mesh interconnected-network**

Keywords—Mesh; Wireless; Network; VPN; LAN

I. Introduction

Mesh networks are growing more and more popular as better and better consumer systems are released. However, a lot of mesh networks focus on connecting already existing LAN network. While this is extreamly useful in it's own right, there can be many more use-cases for a wireless mesh interconnected network. Lets take for example two neighbors that have their own LAN setup who which to commuincate. To do this currently, they must know eachothers public IP, and go through their local ISP. They could join their networks together using very long ethernet cables or by using an ad-hoc wifi network. However, this requires a lot of setup to get running properly, and could be insecure. It also could leave both networks entirely open to each other which is not desired.

This paper proposes a protocol and solution that two neighbors could use to create a device to connect their local LANs. With the proposed protocol, the two neighbors could access each-others LAN without having to expose their entire network and withtout having to do complicated setup. This protocol could also be used to create a expansive interconnected network of Nodes.

In this paper we attempt to solve a very hard problem in Computer Science, and while this proposed solution does not cover all the holes, it points in the right direction on how a wireless interconnected mesh network should function.

II. Problem

A. Connect multiple LANs using mesh

We want a way to connect multiple LANs using a wireless mesh network. Each connected LAN must provide a Node that will act as the Gateway. Each conneted Node that is connected to the Gateway is called a Peer. Nodes will be able to join the mesh using a wireless ad-hoc network using IPv6 addressing

B. Prove authenticity in the mesh network in a decentralized manner

Due to the nature of mesh networks, there is no central authority. This means that each Node must prove their own authenticity whenever they send a packet using the protocol.

C. Solution must work with existing applications and a modern average LAN

We want this solution to be easy to implement in existing LANs and work with existing applications protocls (HTTP, SSH, etc..).

D. Solution must work if a Node isn't connected to a LAN.

In order to make this solution useful, a Node shouldn't have to be connected to a LAN in order to act as a Peer.

III. Wireless Mesh Network Protocol

In order to solve these problems, we must create a protocol that each participating Node must conform to. We cannot define a solution until we define a protocol. The protocol will be built on top of the Application Layer of the OSI Model.

A. Definations

1) *WMN - Wireless Mesh Network, Defines the protocol.*
2) *Node – Any participating device*

3) Gateway – Any node that provides access to the WMN through a LAN

4) Peer – Any Node that is connected to a Node relative to that Node

B. *Preparing to Join the WMN – Addressing*

Before we can join the WMN, we must figure out what our address must be. The WMN has no central authority (i.e DHCP server), so each Node must pick their own address. However, this means any Node can pick the address of any other Node and disguise themselves as another Peer. To prevent this, we must make the addressing native to the protocol, so a Node can't choose another Node's address without breaking the protocol.

Before joining a WMN, the Node should generate a private/public key pair (or load one) that is 128 bits. The 128 bits of the public key will be the IPv6 address of the Node. Before any packet is sent through the WMN, the application layer of the packet is encrypted with the private key and must be decrypted with the public key. Since the public key is the address of the packet, any node can decrypt the packet.

Since only the original Node knows the private key, only the original Node can create the valid encrypted packet with its own address. If any bad actor tires to copy a Peer's address, they would have to know the private key pair in order to create a valid encrypted packet. This will prove the authenticity of any Node in a decentralized manner. This will be important in the next section. From this point forward, any packet sent from a Node will have the source address be the 128 bits of the public key. The application layer of the packet will be encrypted with the private key pair.

C. *Joining the WMN – Handshake Packet*

In order to join the WMN, a Node must first broadcast a handshake packet. The purpose of this handshake packet is to alert Peers within range that there is a new Node joining the network. The handshake packet takes the following format

TABLE I. HANDSHAKE PACKET

Name	***Type***	***# Bits***
Magic	long	64
Checksum	MD5 Hash	128
Connected Peer Count (CPC)	short	16
Peer Addresses	byte[]	128 * CPC

Figure 1: The handshake packet format.

The magic value of the handshake packet is always 0xff1. This is used to identify the packet as a handshake packet. The checksum value is the MD5 hash of the rest of the packet contents. This is used as error correction to verify that the packet was received correctly.

The Peer Addresses byte array is an array that contains every IPv6 address the connecting Peer is aware of. If the Node is joining the network for the first time, then this array will most likely be omitted as the Node will not be aware of any Peers. If the Node is reconnecting from downtime, then it may populate the array with Peers it knew about before. The purpose of sending all known peers is so a Node knows how to route a packet to reach another Peer.

Nodes must respond to the handshake packet in order to acknowledge that they are aware of the Peer. To respond to the handshake packet they must send a handshake packet to the connecting Peer. Nodes can also choose to reject a handshake packet by sending a rejection packet. Nodes can reject a Node for any defined reason. The rejection packet takes the following format.

TABLE II. REJECTION PACKET

Name	***Type***	***# Bits***
Magic	long	64
Checksum	MD5 Hash	128
Reason	short	16

Figure 2: The rejection packet format.

The magic value of the rejection packet is always 0xffa. The Reason value is defined in Figure 3:

TABLE III. REJECTION REASONS

short	***Reason***
101	Error accepting
102	Peer already exists
103	Peer blocked
104	Peer down
105	Not accepting anymore Peers

Figure 3: Valid short values for the Reason field in Figure 2

A Rejection Packet can be sent after any packet to signify you're disconnecting from a Peer. For example, if a Node is shutting down it may send 104 to all Peers to alert to the network that the Node is shutting down. Another example is if a Peer doesn't send a Heartbeat Packet within the timeout window. This is discussed more in section D.

If a Node decides to accept a handshake and respond with its own handshake, it must first add the Peer's IP address to its list of known peers. After two Nodes establish a handshake, they must periodically send a heartbeat packet to each other to show that they are still an active Node.

D. *Informed Availiablity – Heartbeat Packet*

In order to inform other Peers that the Node is still active and accepting packets, the Node must periodically send a heartbeat packet. The heartbeat packet takes the same form as the Handshake packet in Figure 1, however the Magic value is always 0xff2.

If a Node doesn't hear a handshake packet after a certain time interval, it can assume that Peer is down and can remove it from its known list of Peers. In order for the Peer to reconnect, they must send a new handshake packet. Nodes can reject a handshake packet for any of the defined reasons in Figure 2. If a Node gets a rejection after sending a heartbeat packet, then it can assume that the Peer will no longer accept packets. The Node can attempt to reestablish connection by retransmitting a new handshake packet.

E. *Forward Everything – Transmitting and Retransmitting Packets*

As defined in section B, every packet is encrypted with a Node's private key and must be decrypted with the Node's public key. Since we are using radio waves to transmit the packets, any device can listen to packets being transmitted, however only the source Node can transmit packets from itself. Every packet is prepended with 3 fields that defines how to route the packet and who the source Peer is. This is defined as the WMN Layer, which lives on-top of the Application Layer. The WMN Layer is the only layer of the packet that isn't encrypted when sent initially.

To transmit a new packet to the WMN network, the Node must create a WMN Header. The header is defined in the following format:

TABLE IV. WMN HEADER

Name	*Type*	*# Bits*
Source	IPv6 Address	128
Checksum	SHA256	256
Destination	IPv6 Address	128

Figure 4: The WMN Header that is prepended to every packet transmitted through the network.

The Source field in Figure 4 is the Node's 128-bit public key. This field is used to decrypt the Application Layer portion of the packet and verify the authenticity of the packet.

The Checksum field is the SHA256 hash of the **decrypted** packet. A Peer can use this field to verify the authenticity of the packet. The computed SHA256 of the decrypted packet should match the Checksum field. If it doesn't, then the packet contents are not authentic and was not encrypted by the true source Node as defined in the Source field.

The Destination field is the destination Node this packet is meant for. Since any Node can listen to the packets being transmitted within range, Nodes *should* retransmit a packet if it knows the Destination Peer or if one of its Peers knows the Destination Peer. Thus, it should retransmit the packet if the Destination Peer is in the Node's known list of peers as defined in Section C.

IV. LAN IMPLEMENTATION

In order for this Wireless Mesh Network to be useful, it needs to be able to exist with an already existing LAN. We will do this by making the Node act as a Gateway between the LAN and the WMN. This Gateway will have two interfaces and use preexisting protocols already well defined in LANs, such as DHCPv6 [1].

The Gateway will have two interfaces, one Ethernet interface (eth0) and one WiFi interface (wlan0). The Ethernet interface is connected to a LAN and forwards packets from eth0 to wlan0 depending on the destination IP in the packet. The Gateway will have an IP Table that links ULA IPv6 addresses in the LAN to IPv6 addresses of Peers. The IP Table may look something like this:

TABLE V. EXAMPLE GATEWAY IP TABLE

ULA LAN IP	*WMN IP*
fd07:a47c:3742:823e:3b02:76:982b:463/64	2001:4860:4860::8888
fd07:a47c:2742:82af:3c02:86:914a:463/64	2001:4860:4860::8844

Figure 4: An example of a Gateway's IP Table

When a Peer connects to the Node, the Gateway will make a DHCPv6 request to lease a new IPv6 IP in the LAN. If this fails, then a Reject Packet should be sent to the Peer with Reason 101 or 105 from Figure 3. If the DHCP request is successful and the Gateway was able to lease a local IPv6 address within the LAN, then it'll add this to its local IP Table. All packets sent to that IPv6 address is then routed to that Peer. This is done by reconstructing the packet into a WMN Packet. First, the IP Layer is modified and a random source port is chosen to represent the sender in the LAN. This is put into a port table, which may look something like this:

TABLE VI. EXAMPLE GATEWAY PORT TABLE

Source Port	*Source IP*
23	192.168.1.100
473	192.168.1.101

Figure 4: An example of a Gateway's Port Table

This is used to determine who the original sender is in the LAN when a response packet is received in the future. Then, the contents of the packet are put through the SHA256 algorithm to generate the checksum hash. Then the contents of the packet are encrypted with the Node's private key. Finally, the WMN Header is prepended onto the Packet with the Source being the Node's public key, the Checksum field being the result of the SHA256, and the Destination Node being the WMN IP from the IP Table. The packet may be altered by the Node before reconstruction to enable source port mapping and modifying the source/destination IPs in the IP Layer.

When a Gateway receives a packet from the WMN, it will decrypt the packet. It will then obtain the destination port; this destination port will determine who in the LAN is the packet owner. The Gateway at this point can do port forwarding to determine if the packet is for a server (and if the source Peer can access that server), or check its Port Table to see if the packet is for a client talking to a Peer.

A gateway may also respond to a DNS query within the LAN and lease a temporary IPv6 address to resolve Peers it's not directly connected with.

CONCLUSION

This paper proposes a solution to a very tough problem in computer science. There are still many problems with this solution, one being that any Node on the network can listen to and decrypt packets, I do feel it's a start and points in the right direction. The OSI Model was developed in a stack for a reason, and WMN offers a new layer to the current stack.

In order to solve a lot of the scalability problems of the Internet, we must consider a future of a decentralized internet. Creating a Wireless Mesh Network may aid us in creating that future, it may not. It will however help us connect with our neighbors.

REFERENCES

[1] RFC4339: https://tools.ietf.org/html/rfc4339G.